Task 3: Making changes to Studily

Studily is an AI enabled application that helps students who learn from YouTube to not get distracted by watching non-educational video in their recommendation.

How it works:

I used selenium which is a very popular browser automation library in python. Once the application is started, YouTube is opened on a chrome browser and the user is allowed to watch any educational video they wish to see. The program regularly checks the title of the video that the user is watching and uses AI to make a prediction if the title of the video implies Educational or Non-Educational. If it is Non- Educational, the user is redirected to a different URL, thus not allowing them to watch the video.

My task was to make changes to this application:

- a. Add multiple categories, instead of binary, for the models to classify videos under.
- b. Define a minimum confidence threshold for the models to classify videos under if the target prediction is below the threshold, simply raise an exception "unable to categorize"
- c. List out the probabilities for the top 3/top 5 predicted classes.

To start-off, I needed new data to work with. For this I used my youtube data extraction tool which uses YouTube API to get the titles of videos based on keywords provided.

I gathered around 1500 datapoints for the following classes:

- a. Educational
- b. Music
- c. Sports
- d. Movies
- e. Gaming

```
dataset['Category'].value_counts()
```

```
0 1485
```

3 1480

4 1479

1 1477

2 1459

Name: Category, dtype: int64

Now, each datapoint had to be cleaned and converted into TF-IDF vectors. But before that stop words like {"the", "I","a","to"} etc had to be removed. For this I used the natural language toolkit library, in short NLTK which comes with a list of all the stop words. Looping through each word of a title, the stop words were removed.

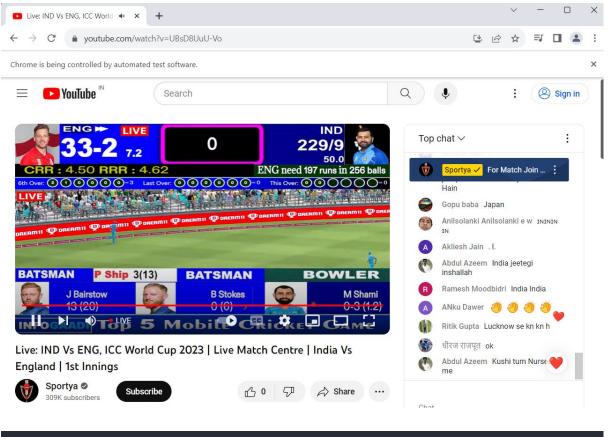
Then the data was converted into TF-IDF vectors. This is important because we need to convert the text into some form of meaningful numbers. TF-IDF basically assigns each word some importance value based on how many times a word has appeared and how many times it is there in each document.

Finally, the vectors were fed into a PassiveAggressiveClassifier. This works by changing its weights and biases aggressively when it gets a prediction wrong during training, and remaining passive when it gets the predictions correct.

	precision	recall	f1-score	support
Educational	0.91	0.90	0.91	377
Music	0.97	0.96	0.97	360
Sports	0.90	0.92	0.91	347
Gaming	0.93	0.95	0.94	376
Movies	0.95	0.92	0.93	385
accuracy			0.93	1845
macro avg	0.93	0.93	0.93	1845
weighted avg	0.93	0.93	0.93	1845

Here is the evaluation report

INTEGRATION WITH YOUTUBE AND SELENIUM



PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

Educational: 15.383166269777565

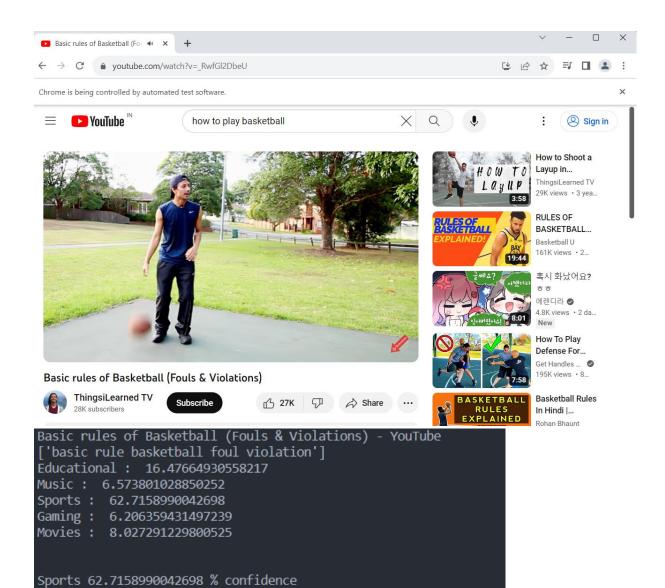
Music: 10.254423874187935

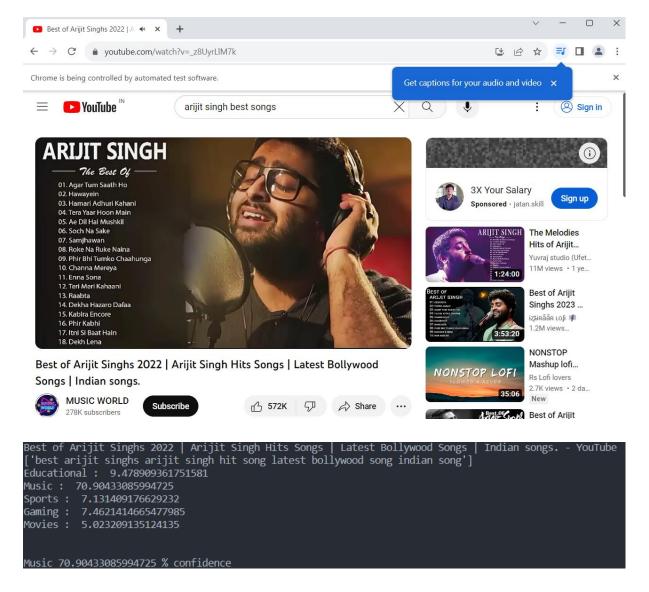
Sports: 48.00212809146668

Gaming: 11.236098244995969

Movies: 15.124183519571838

Sports 48.00212809146668 % confidence





The program is also capable of making predictions on Titles of different languages such as Hindi, german, French, etc

